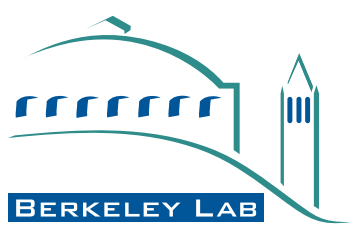
IceCube Software Development Environment

Simon Patton
(LBNL)

Ernest Orlando Lawrence Berkeley National Laboratory



Overview

- **Purpose of this presentation:**
  - Enable a developer to start using the IceCube Software Dev. Env.
- **Contents:**
  - Review what is a Dev. Env.
  - Review benefits of a Dev. Env.
  - Discuss concepts used in Dev. Env.
  - Walk through of Personal Dev. Env.

What is a Development Environment?

- **A set of tools that help create software.**
- **A set of procedures the provide overall organization to a software project.**
- **Can include one or all of the following:**

  Editors, compliers, build system, source code archiving, configuration management, test system, release system, tracking systems.

Benefits of a Development Environment (I)

- **Facilitates efficient communications between developers**
  - communications are easier, faster and less prone to errors.
- **Only need to invest the development time once.**
  - Each developer does not have to create one for themselves.
- **Easier to transition to other parts of the software effort.**

Benefits of a Development Environment (II)

- **Improves Maintainability.**
  - Improves ability to locate problems.
  - Easier to track known problems and their solutions.
  - Maintains “institutional memory.”

Personal Dev. Env.

- **Provides an individual developer with all the tools they need to create, test and submit code that is part of a larger collaborative effort.**
  - This does not exclude its use on non-collaborative efforts.

    [This presentation]

Collaborative Dev. Env.

- **Provides those tools necessary to bring together individual contributions so that they can be tracked, assessed, tested and used to create a deliverable Product.**

  [David G.'s presentation]

Concepts in Personal Dev. Env.

- **Workspace**
  - An area (usually a directory) in which the process of developing a feature or fixing a problem is executed.
- **Project**
  - A collection of classes and resources that are packaged up into a single library file (`.jar` file for Java)
- **Package** ***(not to be confused with Project)***
  - A Java concept which defines a naming and accessibility scope.
- **Standalone**
  - A Workspace without an associated *Baseline*

Concepts in Personal Dev. Env.

(yet to be implemented)

- **Baseline**
  - An agreed set of code and resources that function together.
- **Release**
  - A *Baseline* which has been handed to the “clients”.

Workflow in Personal Dev. Env.

- User creates a *Workspace* in which to work on a specific change.
- Creates tests to verify modifications result in required change.
- Implements code so that tests run successfully.
- Archives changes (tests and code) to software repository.
- *Delivers* changes to be integrated with the rest of the system.

Preparing a Workspace

- **Create directory to be Workspace**

```
[patton@acme]$ mkdir work
```

- **Move into that directory**

```
[patton@acme]$ cd work
```

- **Initialize with `bfd` (baselined file development) system.**

```
[patton@acme]$ bfd init /home/icecube/tools
```

  - This initializes the workspace in *standalone* mode.

- **Set up local environment.**

```
[patton@acme]$ . setup.sh
```

Using Existing Projects

- In standalone mode their are not pre-existing files, so have to build all required files in the local workspace.
- This example uses the following projects:
  - `icebucket`
  - `faye`
  - `icecube-faye`
- These need to be checked out.

```
[patton@acme]$ bfd checkout icebucket
[patton@acme]$ bfd checkout faye
[patton@acme]$ bfd checkout icecube-faye
```

Building a Project

- **Three ways to build a project.**
  - From projects top directory.

```
[patton@acme]$ cd icebucket
[patton@acme]$ ant lib
```

  - Explicitly from the main workspace.

```
[patton@acme]$ ant lib -DPROJECT=icebucket
```

  - Setting it as the default project of the workspace.

```
[patton@acme]$ ant defaultProject -DPROJECT=icebucket
[patton@acme]$ ant lib
```

- **Now need to build all three projects.**

Creating a new Java Project

[It is easier if a new project's directory is created in the archive before any files are created.]

- **Assuming the new project's directory has been created, then need to check it out and create basic contents.**

```
[patton@acme]$ bfd co preston
[patton@acme]$ ant createProject -DPROJECT=preston
  -DPACKAGE=icecube.preston.wool
  -Dpackage_dir=icecube/preston/wool
[patton@acme]$ ant -DPROJECT=preston defaultProject
```

  - option `-Dpackage_dir` is a kludge until we add new `ant` task to map a package on to its directory

Contents of a package

```
preston
    +---+ build.xml
        + project.properties
        + src
        |     +---- icecube
        |               +---- preston
        |                         +---- wool
        |                                   +---+ package.html
        |                                       + test
        |                                         +---- package.html
        + resources
              +---- test
```

Creating a new Java Class (I)

- In this example we are going to create a plug-in for Faye which simply counts events.

```
[patton@acme]$ ant createClass -DCLASS=Counter
```

- Creates two new files in the default package.

```
preston/src/icecube/preston/wool/test/CounterTest.java
preston/src/icecube/preston/wool/Counter.java
```

Creating a new Java Class (II)

- **Fill in tests based on requirements.**

  `preston/src/icecube/preston/wool/test/CounterTest.java`

- **Add dependencies to properties file (based on imports).**

  `preston/src/icecube/preston/project.properties`

- **Fill in interface and implementation of class so tests will succeed.**

  `preston/src/icecube/preston/wool/Counter.java`

- **Update the dependencies.**

  `preston/src/icecube/preston/project.properties`

Building and Testing the new Java Class (II)

- **Now compile new code.**

```
[patton@acme]$ ant lib
```

- **If that succeeds run the tests.**

```
[patton@acme]$ ant test
```

- **If you want an HTML report on the tests.**

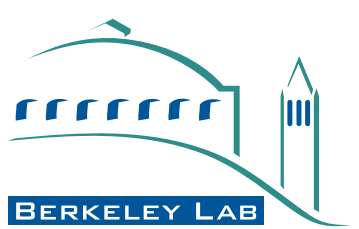
```
[patton@acme]$ ant report
```

  - The resulting report can be accessed via:

```
docs/junit/index.html
```

- **It is the default target so this is the same as the following.**

```
[patton@acme]$ ant
```

Archiving a Project

- Need to specify any new files to be archived.

```
[patton@acme]$ bfd add preston/build.xml
[patton@acme]$ bfd add preston/project.properties
[patton@acme]$ bfd add preston/resources/test
[patton@acme]$ bfd add \
  preston/src/icecube/preston/wool/Counter.Java
[patton@acme]$ bfd add \
  preston/src/icecube/preston/wool/package.html
[patton@acme]$ bfd add \
  preston/src/icecube/preston/wool/test/CounterTest.Java
[patton@acme]$ bfd add \
  preston/src/icecube/preston/wool/test/package.html
```

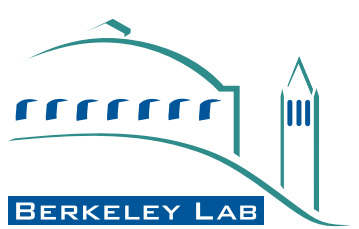
- Then archive the project.

```
[patton@acme]$ bfd archive -m “Project Files” preston
```

Delivering a Project

- **When the development is ready for other to use it should be delivered.**
  - There are three levels of delivery.
    - bug fix [`-b`]
    - minor modifications [`-n`] (public API has changed by feature set has not)
    - Major modifications [`-j`] (feature set has changed)
- **In this case, as the initial features are still under developed, this is a minor modification.**

```
[patton@acme]$ bfd deliver -n preston
```

Summary

- **Reviewed what is a Development Environment and what are its benefits.**
- **Introduced some of the concepts used in a Development Environment.**
- **Walked through an example of the Personal Development Environment and introduced the following:**
  - the workflow used in the environment;
  - the `bfd` system;
  - some common `ant` targets